US-PAT-NO: 6108764

DOCUMENT-IDENTIFIER: US 6108764 A

TITLE: Non-uniform memory access (NUMA) data processing system with multiple caches concurrently holding data in a recent state from which data can be sourced by shared intervention

DATE-ISSUED: August 22, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Baumgartner; Yoanna	Austin	TX	N/A	N/A
Elman; Anna	Austin	TX	N/A	N/A

US-CL-CURRENT: 712/28, 710/104 , 711/119 , 711/122 , 711/130 , 711/141

ABSTRACT:

A non-uniform memory access (NUMA) computer system includes first and second processing nodes that are coupled together. The first processing node includes a system memory and first and second processors that each have a respective associated cache hierarchy. The second processing node includes at least a third processor and a system memory. If the cache hierarchy of the first processor holds an unmodified copy of a cache line and receives a request for the cache line from the third processor, the cache hierarchy of the first processor sources the requested cache line to the third processor and retains a copy of the cache line in a Recent **coherency** state from which the cache hierarchy of the first processor can source the cache line in response to subsequent requests.

10 Claims, 4 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 4

---------- KWIC ---------

Abstract Text - ABTX (1):
A non-uniform memory access (NUMA) computer system includes first and second processing nodes that are coupled together. The first processing node includes a system memory and first and second processors that each have a respective associated cache hierarchy. The second processing node includes at least a third processor and a system memory. If the cache hierarchy of the first processor holds an unmodified copy of a cache line and receives a request for the cache line from the third processor, the cache hierarchy of the first processor sources the requested cache line to the third processor and retains a copy of the cache line in a Recent **coherency** state from which the cache hierarchy of the first processor can source the cache line in response to subsequent requests.

Parent Case Text - PCTX (4):
(2) Ser. No. 09/024,307, "Apparatus and method of maintaining Cache **Coherency** in a multi-processor computer system with Global and Local Recently Read States," which was filed on Feb. 17, 1998, now as U.S. Pat. No. 6,018,791, issued on Jan. 25, 2000, and incorporated herein by reference; and

Brief Summary Text - BSTX (7):
As a result, an MP computer system topology known as non-uniform memory access (NUMA) has emerged as an alternative design that addresses many of the limitations of SMP computer systems at the expense of some additional complexity. A typical NUMA computer system includes a number of interconnected nodes that each include one or more processors and a local "system" memory. Such computer systems are said to have a non-uniform memory access because each processor has lower access latency with respect to data stored in the system memory at its local node than with respect to data stored in the system memory at a remote node. NUMA systems can be further classified as either non-**coherent** or cache **coherent,** depending upon whether or not data **coherency** is

maintained between caches in different nodes. The complexity of cache **coherent**
NUMA (CC-NUMA) systems is attributable in large measure to the additional
communication required for hardware to maintain data **coherency** not only between
the various levels of cache memory and system memory within each node but also
between cache and system memories in different nodes. NUMA computer systems
do, however, address the scalability limitations of conventional SMP computer
systems since each node within a NUMA computer system can be implemented as a
smaller SMP system. Thus, the shared components within each node can be
optimized for use by only a few processors, while the overall system benefits
from the availability of larger scale parallelism while maintaining relatively
low latency.

Brief Summary Text - BSTX (11):

processing node includes at least a third processor and a system memory.
If the cache hierarchy of the first processor holds an unmodified copy of a
cache line and receives a request for the cache line from the third processor,
the cache hierarchy of the first processor sources the requested cache line to
the third processor and retains a copy of the cache line in a Recent **coherency**
state from which the cache hierarchy of the first processor can source the
cache line in response to subsequent requests.

Detailed Description Text - DETX (3):

With reference now to the figures and in particular with reference to FIG.
1, there is depicted an illustrative embodiment of a NUMA computer system in
accordance with the present invention. The depicted embodiment can be
realized, for example, as a workstation, server, or mainframe computer. As
illustrated, NUMA computer system 6 includes a number (N.gtoreq.2) of
processing nodes 8a-8n, which are interconnected by node interconnect 22.
Processing nodes 8a-8n may each include M (M.gtoreq.0) processors 10, a local
interconnect 16, and a system memory 18 that is accessed via a memory
controller 17. Processors 10a-10m are preferably (but not necessarily)
identical and may comprise a processor within the PowerPC.TM. line of
processors available from International Business Machines (IBM)

Corporation of
Armonk, N.Y. In addition to the registers, instruction flow logic and execution units utilized to execute program instructions, which are generally designated as processor core 12, each of processors 10a-10m also includes an on-chip cache hierarchy that is utilized to stage data to the associated processor core 12 from system memories 18. Each cache hierarchy 14 includes at least one level of cache and may include, for example, a level one (L1) cache and a level two (L2) cache having storage capacities of between 8-32 kilobytes (kB) and 1-16 megabytes (MB), respectively. As is conventional, such caches are managed by a cache controller that, among other things, implements a selected cache line replacement scheme and a **coherency** protocol. In the present disclosure, each processor 10 and its associated cache hierarchy 14 is considered to be a single **snooper**.

Detailed Description Text - DETX (4):
Each of processing nodes 8a-8n further includes a respective node controller 20 coupled between local interconnect 16 and node interconnect 22. Each node controller 20 serves as a local agent for remote processing nodes 8 by performing at least two functions. First, each node controller 20 **snoops** the associated local interconnect 16 and facilitates the transmission of local communication transactions (e.g., read requests) to remote processing nodes 8. Second, each node controller 20 **snoops** communication transactions on node interconnect 22 and masters relevant communication transactions on the associated local interconnect 16. Communication on each local interconnect 16 is controlled by an arbiter 24. Arbiters 24 regulate access to local interconnects 16 based on bus request signals generated by processors 10 and compile **coherency** responses for **snooped** communication transactions on local interconnects 16, as discussed further below.

Detailed Description Text - DETX (8):
For purposes of the present discussion, the processing node 8 that stores a particular datum in its system memory 18 is said to be the **home node** for that datum; conversely, others of processing nodes 8a-8n are said to be

remote nodes
with respect to the particular datum.

Detailed Description Text - DETX (9):
Memory **Coherency**

Detailed Description Text - DETX (10):
Because data stored within each system memory 18 can be requested, accessed, and modified by any processor 10 within NUMA computer system 6, NUMA computer system 6 implements a cache **coherence** protocol to maintain **coherence** both between caches in the same processing node and between caches in different processing nodes. Thus, NUMA computer system 6 is properly classified as a CC-NUMA computer system. The specific cache **coherence** protocol that is implemented is implementation-dependent, but in a preferred embodiment of the present invention comprises a variant of the well-known Modified, Exclusive, Shared, Invalid (**MESI**) protocol that includes a fifth R (Recent) state, as discussed in detail in the above-referenced co-pending application. Hereafter, it will be assumed that cache hierarchies 14 and arbiters 24 implement the conventional R-**MESI** protocol, of which node controllers 20 recognize the M, S and I states and consider the E state to be merged into the M state and the R state to be merged into the S state. That is, node controllers 20 assume that data held exclusively by a remote cache has been modified, whether or not the data has actually been modified, and do not distinguish between the S and R states for remotely held data.

Detailed Description Text - DETX (12):
Local interconnects 16 and node interconnect 22 can each be implemented with any bus-based broadcast architecture, switch-based broadcast architecture, or switch-based non-broadcast architecture. However, in a preferred embodiment, at least node interconnect 22 is implemented as a switch-based non-broadcast interconnect governed by the 6xx communication protocol developed by IBM Corporation. Local interconnects 16 and node interconnect 22 permit split

transactions, meaning that no fixed timing relationship exists between the
address and data tenures comprising a communication transaction and that data
packets can be ordered differently than the associated address packets. The
utilization of local interconnects 16 and node interconnect 22 is also
preferably enhanced by pipelining communication transactions, which permits a
subsequent communication transaction to be sourced prior to the master of a
previous communication transaction receiving **coherency** responses from each
recipient.

Detailed Description Text - DETX (13):
Regardless of the type or types of interconnect architecture that are
implemented, at least three types of "packets" (packet being used here
generically to refer to a discrete unit of information)--address, data, and
coherency response--are utilized to convey information between processing nodes
8 via node interconnect 22 and between **snoopers** via local interconnects 16.
Referring now to Tables I and II, a summary of relevant fields and definitions
are given for address and data packets, respectively.

Detailed Description Text - DETX (14):
As indicated in Tables I and II, to permit a recipient node or **snooper** to
determine the communication transaction to which each packet belongs, each
packet in a communication transaction is identified with a transaction tag.
Those skilled in the art will appreciate that additional flow control logic and
associated flow control signals may be utilized to regulate the utilization of
the finite communication resources.

Detailed Description Text - DETX (15):
Within each processing node 8, status and **coherency** responses are
communicated between each **snooper** and the local arbiter 24. The signal lines
within local interconnects 16 that are utilized for status and **coherency**
communication are summarized below in Table III.

Detailed Description Text - DETX (16):
Status and **coherency** responses transmitted via the AResp and AStat

lines of local interconnects 16 preferably have a fixed but programmable timing relationship with the associated address packets. For example, the AStatOut votes, which provide a preliminary indication of whether or not each **snooper** has successfully received an address packet transmitted on local interconnect 16, may be required in the second cycle following receipt of the address packet. Arbiter 24 compiles the AStatOut votes and then issues the AStatIn vote a fixed but programmable number of cycles later (e.g., 1 cycle). Possible AStat votes are summarized below in Table IV.

Detailed Description Text - DETX (17):

Following the AStatIn period, the ARespOut votes may then be required a fixed but programmable number of cycles (e.g., 2 cycles) later. Arbiter 24 also compiles the ARespOut votes of each **snooper** and delivers an ARespIn vote, preferably during the next cycle. The possible AResp votes preferably include the **coherency** responses listed in Table V.

Detailed Description Text - DETX (18):

The ReRun AResp vote, which is usually issued by a node controller 20, indicates that the **snooped** request has a long latency (e.g., the request will be serviced by a processor 10 or system memory 18 at a remote processing node 8) and that the source of the request will be instructed to reissue the transaction at a later time. Thus, in contrast to a Retry AResp vote, a ReRun makes the recipient of a transaction that voted ReRun (and not the originator of the transaction) responsible for causing the communication transaction to be reissued at a later time.

Detailed Description Text - DETX (21):

FIG. 2, each node controller 20, which is coupled between a local interconnect 16 and node interconnect 22, includes a transaction receive unit (TRU) 40, a transaction send unit (TSU) 42, a data receive unit (DRU) 44, and a data send unit (DSU) 46. TRU 40, TSU 42, DRU 44 and DSU 46 can be implemented, for example, with field programmable gate arrays (FPGAs) or application specific integrated circuits (ASICs). As indicated, the address and

data paths
through node controller 20 are bifurcated, with address (and **coherency**) packets
being processed by TRU 40 and TSU 42 and data packets being processed by DSU 44
and DRU 46.

Detailed Description Text - DETX (22):
TRU 40, which is so designated to indicate transaction flow off of node
interconnect 22, is responsible for accepting address and **coherency** packets
from node interconnect 22, issuing transactions on local interconnect 16, and
forwarding responses to TSU 42. TRU 40 includes response multiplexer (mux) 52,
which receives packets from node interconnect 22 and passes selected packets to
both bus master 54 and **coherency** response logic 56 within TSU 42. In response
to receipt of a address packet from response multiplexer 52, bus master 54 can
initiate a communication transaction on its local interconnect 16 that is the
same as or different from the type of communication transaction indicated by
the received address packet.

Detailed Description Text - DETX (23):
TSU 42, which as indicated by its nomenclature is a conduit for transactions
flowing onto node interconnect 22, includes a multiple-entry pending buffer 60
that temporarily stores attributes of communication transactions sourced onto
node interconnect 22 that have yet to be completed. The transaction attributes
stored in an entry of pending buffer 60 preferably include at least the address
(including tag) of the transaction, the type of the transaction, and the number
of expected **coherency** responses. Each pending buffer entry has an associated
status, which can be set either to Null, indicating that the pending buffer
entry can be deleted, or to ReRun, indicating that the transaction is still
pending. In addition to sourcing address packets on node interconnect 22, TSU
42 interacts with TRU 40 to process memory request transactions and issues
commands to DRU 44 and DSU 46 to control the transfer of data between local
interconnect 16 and node interconnect 22. TSU 42 also implements the

selected
(i.e., MSI) **coherency** protocol for node interconnect 22 with **coherency** response
logic 56 and maintains **coherence** directory 50 with directory control logic 58.

Detailed Description Text - DETX (24):
Coherence directory 50 stores indications of the system memory addresses of
data (e.g., cache lines) checked out to caches in remote nodes for which the
local processing node is the **home node**. The address indication for each cache
line is stored in association with an identifier of each remote processing node
having a copy of the cache line and the **coherency** status of the cache line at
each such remote processing node. Possible **coherency** states for entries in
coherency directory 50 are summarized in Table VI.

Detailed Description Text - DETX (25):
As indicated in Table VI, the knowledge of the **coherency** states of cache
lines held by remote processing nodes is imprecise. This imprecision is due to
the fact that a cache line held remotely can make a transition from R, S or E
to I or from E to M without notifying the node controller 20 of the **home node**.

Detailed Description Text - DETX (27):
In order to decrease latency of processor read requests, the present
invention supports shared intervention, that is, the sourcing of data in
response to a read request by a cache holding data in an unmodified (i.e., E or
M) state, in NUMA computer system 6. Because multiple caches in NUMA computer
system 6 may concurrently hold the same unmodified cache line, some mechanism
is required to regulate which cache sources the cache line by shared
intervention. As described in the above-referenced co-pending applications,
that mechanism is the R (Recent) cache **coherency** state. In accordance with the
present invention, only one cache hierarchy 14 in a particular processing node
8 can hold a particular cache line in the R state at any one time; however,
cache hierarchies 14 in multiple processing nodes 8 may concurrently hold the
same cache line in the R state.

Detailed Description Text - DETX (29):
As indicated, if cache hierarchy 14 receives an ARespIn Shared **coherency** vote, the cache controller of cache hierarchy 14 "knows" that no other **snooper** in the same processing node 8 holds the requested cache line in R state or M state and that the requested cache line will be supplied by either the local system memory 18 or a remote system memory 18 via node controller 20. Accordingly, when requesting cache hierarchy 14 receives the requested cache line via local interconnect 16, the cache controller of cache hierarchy 14 caches the requested cache line and sets its **coherency** state to Recent, meaning that, of the multiple local cache hierarchies 14 holding the requested cache line, the requesting cache hierarchy 14 is responsible for sourcing the requested cache line by Shared intervention.

Detailed Description Text - DETX (30):
If the requesting cache hierarchy 14 receives an ARespIn Null **coherency** vote in response to the read request, the cache controller of the requesting cache hierarchy 14 "knows" that no local cache hierarchy 14 stores a copy of the requested cache line and that the requested cache line will be sourced by either the local system memory 18 or a remote system memory via node controller 20. When the requested cache line is received by requesting cache hierarchy 14, the requested cache line is cached in the Exclusive state.

Detailed Description Text - DETX (31):
If the requesting cache hierarchy 14 receives an ARespIn Shared intervention or Modified intervention vote, the cache controller at requesting processor 10 "knows" that the requested cache line will be sourced by another **snooper** in the same processing node 10, and upon receipt of the requested cache line stores it in the R state.

Detailed Description Text - DETX (32):
The state transitions within the cache hierarchy of a **snooper** in response to receipt of a read request are summarized below in Table VIII.

Importantly, the influence of a read request on the **coherency** state of a cache line depends upon whether the read request was received from a local processor 10 or was received from a processor 10 in a remote processing node 8 via the local node controller 20. The information regarding the source of the read request can be conveyed to **snooping** processors 10 in a number of ways. For example, node controller 20 can supply a "remote request" signal to the cache hierarchy 14 of each processor 10 that indicates when node controller 20 has sourced a read request from a remote processing node 8 on the local interconnect 16. Such a "remote request" signal can be inserted into a defined (e.g., transaction type) field within the read request transaction on local interconnect 16 by node controller 20 or can be transmitted via a separate signal line connecting node controller 20 to each processor 10.

Detailed Description Text - DETX (33):

As shown in Table VIII, if a cache hierarchy 14 **snoops** a read request issued by a local processor 10 (i.e., the "remote request" signal is not asserted) and holds the requested cache line in either Exclusive state or Recent state, the **snooping** cache hierarchy 14 provides a Shared intervention ARespOut vote, sources the requested cache line on local interconnect 16 in response to receipt of a Shared intervention ARespIn vote from arbiter 24, and updates the **coherency** state of its copy of the requested cache line to Shared state. Similarly, if a cache hierarchy 14 **snoops** a read request issued by a local processor 10 and holds the requested cache line in Modified state, the **snooping** cache hierarchy 14 provides a Modified intervention ARespOut vote, sources the requested cache line on local interconnect 16 in response to receipt of a Modified intervention ARespIn vote, and updates the **coherency** state of its copy of the requested cache line to Shared state. If, on the other hand, a **snooping** cache hierarchy 14 holds a cache line requested by a local or remote processor 10 in Shared or Invalid state, the **snooping** cache hierarchy 14 supplies the

appropriate ARespOut vote (i.e., Shared or Null, respectively), but does not source data.

Detailed Description Text - DETX (34):

The remaining three cases shown in Table VIII occur when a **snooping** cache hierarchy 14 at the home or remote node of a cache line receives a read request for the cache line from a remote processing node 8 via the local node controller 20. As noted above, such read requests are identified by a "remote request" signal. In response to receipt of such a read request, the **snooping** cache hierarchy 14 supplies the appropriate ARespOut vote, namely, Shared intervention if the requested cache line is held in either the Exclusive or Recent state and Modified intervention if the requested cache line is held in the Modified state. Then, in response to receipt of an ARespin Shared intervention signal (if an ARespout Shared intervention vote was given) or ARespIn Modified intervention signal (if an ARespOut Modified intervention vote was given), the **snooping** cache hierarchy 14 sources the requested cache line on local interconnect 16. In addition, the **coherency** state of the requested cache line at the **snooping** cache hierarchy 14 is updated to the Recent state, if in Exclusive or Modified state, and remains unchanged if already set to the Recent state. The cache line sourced on local interconnect 16 by the **snooping** cache hierarchy 14 is received by the local controller 20, which transmits the cache line to the node controller of the requesting processing node 8 via node interconnect 22.

Detailed Description Text - DETX (35):

For those states and operations not shown in Tables VII and VIII, **coherency** state transitions and **coherency** responses are performed in accordance with the prior art **MESI** protocol, with the Recent state being treated like the Shared state.

Detailed Description Text - DETX (37):

As shown in FIG. 3A, processor 10b of processing node 8b first requests a

cache line from its cache hierarchy 14 that has processing node 8a as the **home** **node**. In response to the request missing in cache hierarchy 14, processor 10b sources a read request for the cache line on its local interconnect 16.

Processor 10a **snoops** the read request and responds with a Null ARespOut vote, indicating that cache hierarchy 14 of processor 10a does not store a copy of the requested cache line, and node controller 20 votes ARespOut ReRun. In response to the arbiter (not illustrated) compiling the ARespOut votes and voting ARespIn ReRun, node controller 20 forwards the read request to the **home** **node,** i.e., processing node 8a, via node interconnect 22.

Detailed Description Text - DETX (38):
In response to receipt of the forwarded read **request, node** controller 20 of processing node 8a forwards the read request onto local interconnect 16 of processing node 8a in conjunction with a "remote request" signal. Because cache hierarchy 14 of processor 10b stores a copy of the requested cache line in the Recent state, processor 10b provides an ARespOut Shared intervention vote. Processor 10b subsequently sources the requested cache line onto local interconnect 16 by shared intervention in response to receipt of an ARespIn Shared intervention vote from the arbiter, as shown in FIG. 3B. However, because processor 10b was notified by node controller 20 that the read request was a "remote request" forwarded from processing node 8b, cache hierarchy 14 of processor 10b retains the requested cache line in Recent state.

Detailed Description Text - DETX (39):
The data transaction containing the requested cache line is received by node controller 20 of processing node 8a and forwarded to node controller 20 of processing node 8b via node interconnect 22. **Coherency** directory 50 of processing node 8a records that the cache line has been "checked out" to processing node 8b non-exclusively by updating the **coherency** state of the cache line to the imprecise Shared state. The requested cache line is then supplied by node controller 20 of processing node 8b to processor 10b, which

stores the requested cache line in the Recent state. In this manner, processor 10b at each of processing nodes 8a and 8b may

Detailed Description Text - DETX (41):
As has been described, the present invention provides a NUMA computer system that advantageously reduces the number of inter-node read requests and reduces the latency of read requests by supporting shared intervention of data.

According to the present invention, each processing node may have a **snooper** (e.g., cache hierarchy) that holds the same cache line in a non-exclusive Recent state from which that **snooper** can source the cache line. To prevent **snoopers** from updating the **coherency** state of a cache line held in Recent, Exclusive, or Modified state to Shared state in response to a remote processor's read request, each **snooper** is notified of such requests by a remote request signal.

Detailed Description Paragraph Table - DETL (1):
TABLE I ______________________________________ Field Name Description ______________________________________ Address Modifiers defining attributes of a <0:7> communication transaction for **coherency**, write thru, and protection Address Tag used to identify all packets within a <8:15> communication transaction Address Address portion that indicates the <16:63> physical, virtual or I/O address in a request AParity Indicates parity for address bits <0:63> <0:2> TDescriptors Indicate size and type of communication transaction.

____

Detailed Description Paragraph Table - DETL (3):
TABLE III ______________________________________ Signal Name Description ______________________________________ AStatOut Encoded signals asserted by each bus receiver to indicate flow control or error information to arbiter AStatIn Encoded signals asserted by arbiter in response to tallying the AStatOut signals asserted by the bus receivers ARespOut Encoded signals

asserted by each bus receiver to indicate **coherency** information to
arbiter
ARespIn Encoded signals asserted by arbiter in response to tallying
the
ARespOut signals asserted by the bus receivers

Detailed Description Paragraph Table - DETL (4):
TABLE IV ______________________________ AStat vote Meaning

______________________________ Null Idle Ack Transaction
accepted by
snooper Error Parity error detected in transaction Retry Retry
transaction,
usually for flow control ______________________________

Detailed Description Paragraph Table - DETL (5):
TABLE V ______________________________ **Coherency**
responses
Meaning ______________________________ Retry Source of request
must
retry transaction usually for flow control reasons Modified Line is
modified
in cache and will be intervention sourced from cache to requestor
Shared Line
is unmodified in cache (and intervention possibly shared) and will be
sourced
from cache to requestor Shared Line is held shared in cache Null Line
is
invalid in cache ReRun **Snooped** request has long latency and source of
request
will be instructed to reissue transaction at a later time

Detailed Description Paragraph Table - DETL (6):
TABLE VI ______________________________ Possible Possible
state(s) **Coherence** state(s) in directory in local remote state cache
cache
Meaning ______________________________ Modified I M,E, or
Cache line
may be (M) I modified at a remote node with respect to system memory
at home
node Shared R, S or R, S or Cache line may be held (S) I I
non-exclusively at
remote node Invalid R,M,E,S, I Cache line is not held (I) or I by any
remote
node Pending- R, S or R, S or Cache line is in the shared I I process
of
being invalidated at remote nodes Pending- I M,E, or Cache line,
which may
modified I be modified remotely, is in process of being written back
to
system memory at **home node,** possibly with invalidation at remote

node

Detailed Description Paragraph Table - DETL (7):
TABLE VII ______________________________ Master state
ARespIn vote
transition received Data source ______________________________
I.fwdarw.R Shared system memory or node cohtroller I.fwdarw.E Null
system
memory or node controller I.fwdarw.R Shared **snooper** intervention
I.fwdarw.R
Modified **snooper** intervention ______________________________

Detailed Description Paragraph Table - DETL (8):
TABLE VIII ______________________________ **Snooper** state
or state
Snooper Read request transition ARespOut vote source
______________________________ I Null local processor or node
controller E.fwdarw.S Shared int. local processor M.fwdarw.S Modified
int.
local processor S Shared local processor or node controller
R.fwdarw.S
Shared int. local processor E.fwdarw.R Shared int. node controller
M.fwdarw.R
Modified int. node controller R Shared int. node controller

Claims Text - CLTX (3):
wherein said first cache hierarchy, responsive to receipt of a read
request
from the second processing node for a copy of an unmodified cache line
held by
said first cache hierarchy, sources a copy of said unmodified cache
line to
said third cache hierarchy and retains said unmodified cache line in a
Recent
coherency state from which said first cache hierarchy can source said
cache
line; and

Claims Text - CLTX (5):
2. The computer system of claim 1, wherein said first cache
hierarchy,
responsive to receipt of a request by said third cache hierarchy for a
copy of
a cache line that said first cache hierarchy holds in modified state,
said
first cache hierarchy sources a copy of said modified cache line to
said third
cache hierarchy and retains said cache line in said Recent **coherency**
state.

Claims Text - CLTX (7):
said first cache hierarchy retains said cache line in said Recent
coherency
state only in response to receipt of an indication that said read
request is
from another processing node.

Claims Text - CLTX (8):
4. The computer system of claim 1, wherein prior to receipt of said
read
request, said first cache hierarchy associates said unmodified cache
line with
a **coherency** state that is one of Exclusive and Recent.

Claims Text - CLTX (12):
if said requested cache line is held in said first cache hierarchy
in an
unmodified state, sourcing a copy of said unmodified cache line from
said first
cache hierarchy to said third cache hierarchy, retaining said cache
line in
said first cache hierarchy in a Recent **coherency** state from which said
first
cache hierarchy can source said cache line, and concurrently storing
said copy
of said cache line in said third cache hierarchy in said Recent
coherency state
to also permit said third cache hierarchy to source said cache line.

Claims Text - CLTX (14):
if said requested cache line is held in said first cache hierarchy
in a
modified state, sourcing a copy of said cache line from said first
cache
hierarchy to said third cache hierarchy and retaining said cache line
in said
first cache hierarchy in said Recent **coherency** state.

Claims Text - CLTX (15):
8. The method of claim 6, wherein retaining said cache line in said
Recent
coherency state comprises retaining said cache line in said Recent
coherency
state only in response to receipt of an indication that said read
request is
from another processing node.

Claims Text - CLTX (17):
prior to receipt of said read request, associating said unmodified
cache

line with a **coherency** state that is one of Exclusive and Recent in said
first
cache hierarchy.